

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平8-263325

(43)公開日 平成8年(1996)10月11日

(51)Int.Cl.⁶

G 0 6 F 11/30

識別記号

3 1 0

庁内整理番号

7313-5B

F I

G 0 6 F 11/30

技術表示箇所

3 1 0 A

審査請求 未請求 請求項の数15 OL (全 12 頁)

(21)出願番号 特願平7-60488

(22)出願日 平成7年(1995)3月20日

(71)出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72)発明者 大橋 勝之

神奈川県川崎市中原区上小田中1015番地

富士通株式会社内

(74)代理人 弁理士 大菅 義之 (外1名)

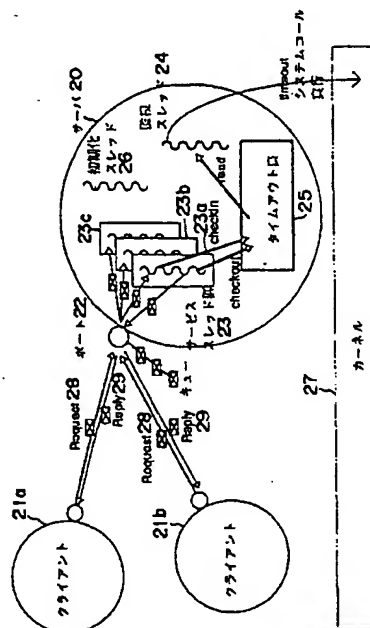
(54)【発明の名称】 サーバ処理装置、サーバ内障害検出装置及びサーバ内障害検出方法

(57)【要約】

【目的】 本発明は、クライアント／サーバモデルのシステムに関し、レスポンスを低下させることなく、複数のクライアントからのリクエストを処理することが可能なサーバ内の障害検出方式を提供することを目的とする。

【構成】 サービススレッド23aは、クライアント21aからのリクエスト28を受信すると、要求された処理のタイムアウト時間と自分のスレッドIDをタイムアウト表25に書込む。そして、リクエスト28の処理を終了すると、タイムアウト表25内のタイムアウト時間及びスレッドIDを消去する。監視スレッド24は、定期的にタイムアウト表25からタイムアウト時間を読み出して、現在の時刻と比較することにより、設定されたタイムアウト時間までに処理を終了していないサービススレッドを検出する。タイムアウトしたサービススレッドを発見した場合には、該サービススレッドを強制的に終了させて、クライアント21にエラーを通知する。

本発明の一実施例の概要を説明する図



【特許請求の範囲】

【請求項1】 クライアントから依頼される処理を、複数のスレッドで処理し、その処理結果を該クライアントに返すサーバ処理装置において、

各スレッドによって実行される処理のタイムアウト時間の管理情報が設定されるタイムアウト時間情報管理手段と、

該各スレッドが処理を開始する際に、該タイムアウト時間情報管理手段にそのスレッドの当該タイムアウト時間情報管理情報を設定するタイムアウト時間情報設定手段と、

上記タイムアウト時間情報管理手段に設定されたタイムアウト時間管理情報を用いて、タイムアウトの発生したスレッドを検出するタイムアウト検出手段と、

を備えることを特徴とするサーバ処理装置。

【請求項2】 前記タイムアウト検出手段は、タイムアウトの発生したスレッドを削除すること、

を特徴とする請求項1記載のサーバ処理装置。

【請求項3】 前記タイムアウト検出手段は、スレッドを削除した後に、新たなスレッドを生成すること、

を特徴とする請求項2記載のサーバ処理装置。

【請求項4】 前記タイムアウト検出手段は、タイムアウトの発生したスレッドを強制的に終了させること、

を特徴とする請求項1記載のサーバ処理装置。

【請求項5】 クライアントからのリクエストを受信し、該リクエストの処理を実行するサービス手段と、

該サービス手段が行う処理のタイムアウト時間を管理するタイムアウト時間管理手段と、

前記サービス手段を監視する監視手段とから構成され、前記サービス手段は、クライアントからリクエストされた処理を開始する前に、該処理のタイムアウト時間を前記タイムアウト時間管理手段に設定し、該処理の終了時にタイムアウト時間管理手段の自分が設定した前記タイムアウト時間を削除し、

前記監視手段は、タイムアウト時間管理手段を参照し、タイムアウト時間内に前記処理が終了しているか否かを検出することを特徴とするサーバ内障害検出装置。

【請求項6】 前記監視手段は、前記検出処理を開始する前にタイムアウト処理を設定し、

該検出処理終了後、該タイムアウト処理を解除すること

を特徴とする請求項5、6又は7記載のサーバ内障害検出装置。

【請求項9】 前記タイムアウト時間管理手段が、スレッドIDを格納する「スレッドID」フィールドおよびタイムアウト時間を設定する「タイムアウト時間」フィールドを有する表から構成されることを特徴とする請求項5、6、7又は8記載のサーバ内障害検出装置。

【請求項10】 前記サービス手段および前記監視手段がプロセスによって構成されることを特徴とする請求項5記載のサーバ内障害検出装置。

【請求項11】 クライアントからのリクエストを受信するポートと、

該リクエストを処理するサービススレッド群と、スレッド番号とタイムアウト時間を対応づけて格納するタイムアウト表と、

サービススレッド群を監視する監視スレッドから構成されるサーバを有し、

サービススレッド群の各サービススレッドは、前記リクエストされた処理を開始する前に、該処理のタイムアウト時間と自分のスレッド番号をタイムアウト表に記入し、該処理の終了後にタイムアウト表から該タイムアウト時間と自分のスレッド番号を削除し、

監視スレッドが定期的にタイムアウト表を読み出して、タイムアウト時間内に処理を終了していないサービススレッドを検出することを特徴とするサーバ内障害検出装置。

【請求項12】 クライアントからのリクエストを受信し、

該受信したリクエストに対する処理を実行するサービススレッドのスレッド番号とタイムアウト時間をタイムアウト表に設定し、

前記リクエストに対する処理を実行し、該処理終了後、前記タイムアウト表に設定されたスレッド番号とタイムアウト時間を削除し、

前記タイムアウト表を参照することによりサービススレッドの監視処理を実行することを特徴とするサーバ内障害検出方法。

【請求項13】 前記監視処理を開始する前に、監視処理のタイムアウト処理を設定し、

前記監視処理の終了後、該タイムアウト処理を解除することを特徴とする請求項12記載のサーバ内障害検出方法。

【請求項14】 前記監視処理を終了後、次の監視処理を始める前に、該監視処理を所定の時間スリープさせることを特徴とする請求項12又は13記載のサーバ内障害検出方法。

【請求項15】 サーバが起動時に、初期化スレッドを生成し、

該初期化スレッドが、タイムアウト表を初期化し、前記

3

監視処理を行う監視スレッドを生成し、クライアントからリクエストを受け付けるリクエスト受付ポートを作成し、該リクエストに対する処理を行うサービススレッド群を生成することを特徴とする請求項12、13又は14記載のサーバ内障害検出方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は計算機上のクライアント／サーバモデルにおけるサーバ内での障害を検出する装置および方法に係わり、特にクライアントから要求されたサービスの処理中にサーバ内で発生した無限ウェイトや無限ループ等の障害を検出する装置および方法に関する。

【0002】

【従来の技術】クライアント／サーバモデルは計算機環境の分散化／並列化のために有効な技術であり、関連する技術の開発が望まれている。クライアント／サーバモデルでは、サーバはクライアントからの要求（リクエスト）を受けてサービス処理を行い、その結果をクライアントに返すよう構成される。

【0003】サーバは、クライアントからのリクエスト処理中に無限ウェイトあるいは無限ループに陥る可能性がある。この場合、サーバだけではなく、サーバからの返答（リプライ）を待っているクライアントも無限ウェイトに陥ってしまい、システム全体が停止してしまう。このような事態が発生するのを避けるため、サーバは、リクエストに対するサービス処理開始前に、システムカーネルがシステムコールとして提供するタイマルーチンを使って該サービス処理のタイムアウト時間を設定していた。サーバが設定したタイムアウト時間内に前記サービス処理が終了しない場合には、システムカーネルがサーバの処理を終了させ、クライアントにサーバのエラーを通知していた。このようにして、サーバの障害がクライアントに伝播してシステム全体が停止する事態を引き起こされるのを防いでいた。

【0004】図9は、上述した従来技術を説明する図である。同図に示すように、サーバ（サーバプロセス）10は、クライアント11aまたはクライアント11bからのリクエスト18を受けてサービス処理を行い、返答（リプライ）19をクライアント11aまたはクライアント11bに返す。リクエスト18およびリプライ19は、メッセージによって行われる。同図において、中に×が書かれた長方形がメッセージを示している。

【0005】図9において、クライアント11aまたはクライアント11bからのリクエスト18は、サーバ10のポート12で受信され、そのポート12のキュー9に入れられる。サービススレッド群13の各サービススレッド13a～13cは、適宜、キュー9からリクエスト18を取り出して、サービス処理を行う。サービス処理が終了すると、各サービススレッド13a～13c

4

は、リプライ19をクライアント11aまたはクライアント11bに返す。

【0006】各サービススレッド13a～13cは、前記サービス処理を開始する前にタイムアウト処理のためのシステムコールを実行する。該システムコールを受けて、カーネル17はタイムアウト処理を行う。すなわち、指定された時間内にサービス処理が終了していない場合は、前記システムコール時に指定される処理ルーチンによって、例えば、タイムアウトしたサービススレッドを消滅させ、リクエスト処理を実行していたクライアントヘエラーを通知する等の処理を行う。

【0007】

【発明が解決しようとする課題】しかしながら、従来の方式では、サーバがクライアントからのリクエストの処理を開始する毎にシステムコールであるタイマルーチンをコールするため、複数のクライアントからの要求を複数のサービススレッドによって並列に処理する場合には、リクエストの増加に伴いシステムコールの実行回数が増加する。システムコールは、サーバ内部の関数呼出しにくらべてオーバーヘッドが大きく、システムコールの実行回数が増加すると、サーバのレスポンスが低下してしまう可能性がある。

【0008】本発明は、サーバ内のサービススレッド数が増加しても、サーバのレスポンス性能の低下を防ぐことができるサーバ内障害検出装置及び方法を提供することを目的とする。

【0009】

【課題を解決するための手段】図1は、本発明の原理を説明する図（その1）である。この発明に係るサーバ処理装置（以下、第1の発明）は、クライアントから依頼される処理を、複数のスレッドで処理し、その処理結果を該クライアントに返すサーバ処理装置を前提とし、以下の各手段を備える。

【0010】タイムアウト時間情報管理手段31は、各スレッドによって実行される処理のタイムアウト時間の管理情報が設定される。これは、例えば、メモリ上にテーブルとして作成される。

【0011】タイムアウト時間情報設定手段33は、各スレッドが処理を開始する際に、タイムアウト時間情報管理手段31にそのスレッドの当該タイムアウト時間管理情報を設定する。

【0012】タイムアウト検出手段35は、上記タイムアウト時間情報管理手段31に設定されたタイムアウト時間管理情報を用いて、タイムアウトの発生したスレッドを検出する。

【0013】上記構成において、前記タイムアウト検出手段35は、例えば、タイムアウトの発生したスレッドを削除するような構成にして良い。また、さらに、該スレッドを削除した後に、新たなスレッドを生成するような構成にしても良い。

【0014】また、前記タイムアウト検出手段35は、例えば、タイムアウトの発生したスレッドを強制的に終了させるような構成にしても良い。図2は、本発明の原理を説明する図（その2）である。

【0015】この発明に係るサーバ内障害検出装置（以下、第2の発明）は、以下の各手段を備える。サービス手段100は、クライアントからのリクエストを受信し、該リクエストに対する処理を実行する。タイムアウト時間管理手段101は、前記サービス手段100が行う処理のタイムアウト時間を管理する。監視手段102

は、前記サービス手段100を監視する。
【0016】前記サービス手段100は、クライアントからのリクエストに対する処理を開始する前に、該処理のタイムアウト時間を前記タイムアウト時間管理手段101に設定し、該処理の終了時にタイムアウト時間管理手段101の自分が設定したタイムアウト時間を削除する。前記監視手段102は、定期的にタイムアウト時間管理手段101を参照し、タイムアウト時間内に前記処理が終了しているか否かを検出する。

【0017】該監視手段102は、前記検出処理を開始する前に、例えば、システムカーネルに依頼して自分自身のタイムアウト処理を設定する。そして、該検出処理終了後、該タイムアウト処理を解除する。

【0018】前記サービス手段100は、例えば、複数のサービススレッドで構成され、前記監視手段102は、例えば、1つの監視スレッドから構成される。また、前記サービス手段100および前記監視手段102をプロセスによって構成するようにしてもよい。

【0019】前記タイムアウト時間管理手段101は、例えば、スレッドIDを格納する「スレッドID」フィールドおよびタイムアウト時間を設定する「タイムアウト時間」フィールドを有する表から構成される。

【0020】本発明に係るサーバ内障害検出方法は、まず、クライアントからのリクエストを受信する。次に、受信したリクエストに対する処理を実行するサービススレッドのスレッド番号とタイムアウト時間をタイムアウト表に設定する。そして、前記リクエストに対する処理を実行する。該処理終了後、前記タイムアウト表に設定されたスレッド番号とタイムアウト時間を削除する。また、前記タイムアウト表を定期的に参照し、タイムアウトしたサービススレッドが存在するか否かを判別することによってサービススレッドの監視処理を実行する。

【0021】また、本発明に係るサーバ内障害検出方法は、前記監視処理を開始する前に、監視処理のタイムアウト処理を設定し、前記監視処理の終了後、該タイムアウト処理を解除するようにする。

【0022】更に、本発明に係るサーバ内障害検出方法は、前記監視処理を終了後、次の監視処理を始める前に、該監視処理を所定の時間スリープさせる。更に、本発明に係るサーバ内障害検出方法は、サーバが起動時

に、初期化スレッドを生成し、該初期化スレッドが、前記タイムアウト表を初期化し、前記監視処理を行う監視スレッドを生成し、クライアントからリクエストを受け付けるリクエスト受付ポートを作成し、該リクエストに対する処理を行うサービススレッド群を生成する。

【0023】

【作用】第1の発明では、タイムアウト時間情報設定手段33が、各スレッドが処理を開始する際に、そのスレッドの当該タイムアウト時間情報管理情報を、タイムアウト時間情報管理手段31に設定する。

【0024】タイムアウト検出手段35は、上記タイムアウト時間情報管理手段31に設定されたタイムアウト時間管理情報を用いて、タイムアウトの発生したスレッドを検出する。

【0025】したがって、スレッドが処理を開始する毎に、カーネルに対してタイマ処理用のシステムコールを発行せずに、該スレッドのタイムアウトの検出することが可能になり、システムコール実行のためのオーバーヘッドが減少する。このため、クライアントに対するサーバのレスポンスが向上する。

【0026】第2の発明では、前記サービス手段100は、リクエストに対する処理を開始する前に、該処理のタイムアウト時間をタイムアウト時間管理手段101に設定し、前記処理を終了すると、タイムアウト時間管理手段101の自分が設定したタイムアウト時間を削除する。

【0027】これにより、前記監視手段102は、定期的にタイムアウト時間管理手段101を参照し、例えば、設定されているタイムアウト時間と、現在の時刻を比較することにより、前記処理が設定されたタイムアウト時間内に終了しているか否かを検出することができる。したがって、サービス手段100が、リクエストに対する処理を開始する毎に、タイムアウト処理の設定をシステムカーネルに対するシステムコールによって行う必要がなくなり、システムコールの実行回数を減らせる。

【0028】

【実施例】以下、本発明の実施例について、図面を参照しながら詳細に説明する。図3は、本発明の一実施例の概略を説明する図である。同図に示すように、本実施例のサーバ（サーバプロセス）20は、クライアント21aまたは21b（以下クライアント21）からのリクエスト28を受け付けるポート22と、クライアントからのリクエスト28を処理するサービススレッド群23と、サービススレッド群23の動作を監視する監視スレッド24と、各サービススレッド23a～23cによってタイムアウト時間が記入されるタイムアウト表25と、サーバ20の初期化を行う初期化スレッド26から構成される。

【0029】本実施例では、複数のサービススレッド2

3a~23cを用意することにより、複数のクライアントからのリクエストを並行して処理することができる。なお、本実施例では、サービススレッド群23は、3つのサービススレッド23a~23cから構成されるが、サービススレッドの数は、サーバ20が受け付けるリクエストの個数に応じて動的に増減させることも可能であり、また、システム構成等に応じて変えるようにしてもよい。

【0030】本実施例においては、オペレーティングシステム(OS)に、マイクロカーネル27でOSの基本的な機能を実現し、このマイクロカーネル27上にマイクロカーネル27が提供するプリミティブなインタフェースを使用して、例えば、UNIXをはじめとする標準的なOSをサブシステムサーバとして複数実装するマイクロカーネル方式を採用している。前記サーバ20は、このサブシステムサーバの1つの例であり、マイクロカーネル27上でサーバとして動作するプロセスである。

【0031】ここで、スレッドとプロセスについて、サービススレッド23a~23cとサーバ20を例にして説明する。サービススレッド23a~23cは、プロセス割当ての最小単位であり、マイクロカーネル27がスケジューリングの対象とする単位である。これに対して、サーバ20は、サービススレッド23a~23cが参照する仮想記憶空間やファイルなどのシステム資源を割り当てる単位である。1個のサーバ20は1個以上のサービススレッドをもつ。

【0032】ポート22は、クライアント21がサービススレッド群23間の通信のためのチャネルであり、例えば、マイクロカーネル27の中に存在する、メッセージのキュー30として実現されている。すなわち、例えば、クライアント21から送られてきたリクエスト28でまだサービススレッド23a~23cによって処理されていないポート22宛のリクエスト28は該キュー30に保持される。

【0033】図3において、サーバ20のポート22へ届いたクライアント21からのリクエスト28はサービススレッド群23によって処理される。サービススレッド群23の各サービススレッド23a~23cは、クライアント21からのリクエスト28を受信すると、要求された処理のタイムアウト時間と自分のスレッド番号(スレッドID)をタイムアウト表25に書込む。各サービススレッド23a~23cは、クライアント21からのリクエスト28の処理を終了すると、ポート22を介してクライアント21へリプライ29を送信する。そして、タイムアウト表25内の自身が設定したタイムアウト時間及びスレッドIDを消去する。

【0034】図4は、タイムアウト表25の一構成例を示す図である。同図に示すように、タイムアウト表は、「スレッドID」フィールド25aと、「タイムアウト時間」フィールド25bから構成される。「スレッドID」

フィールド25aには、各サービススレッド23a~23cを特定するための識別子であるスレッドIDが格納される。「タイムアウト時間」フィールド25bには、タイムアウト時間として、例えば、タイムアウトする時刻を設定する。これは、サービススレッド23a~23cが、「タイムアウト時間」フィールド25bを設定する際、現在の時刻(設定時の時刻)に所定の時間(タイムアウト時間)を加えたものを設定することで行える。現在の時刻は、例えば、不図示のリアル・タイム・クロック等から成るシステム時計から読み出す。

【0035】また、サービススレッド23a~23cは、クライアント21からのリクエスト28の処理を終了すると、「スレッドID」フィールド25aに自身が設定したスレッドIDと、「タイムアウト時間」フィールド25bに自身が設定したタイムアウト時間の2つの情報を消去する。

【0036】図3において、監視スレッド24は、定期的にタイムアウト表25からサービススレッド群23の各サービススレッド23a~23cが設定したタイムアウト時間を読み出して、現在の時刻と比較することにより、設定されたタイムアウト時間までに処理を終了していないスレッドを検出する。監視スレッド24がタイムアウトしたサービススレッドを発見した場合には、該サービススレッドを強制的に終了または消滅させて、クライアント21にエラーを通知する。

【0037】監視スレッド24は、タイムアウトしたサービススレッドの検出処理を行う前に、マイクロカーネル27がシステムコールとして提供するタイマルーチンを使って、自身のタイムアウト時間を設定する。これによって、監視スレッド24自身が無限ウェイトや無限ループに陥ることを避ける。

【0038】次に、上記各スレッドの処理手順を図5~図7に示すフローチャートを参照しながら説明する。なお、以下に示すフローチャートの説明中、S1、S2、・・・は処理手順(ステップ)の番号を示す。

【0039】図5は実施例における初期化スレッド26の処理手順を説明するフローチャートである。マイクロカーネル27によって、サーバ20が起動されると、初期化スレッド26が生成される。サーバ20によって生成された初期化スレッド26は、まず、タイムアウト表25を初期化する(S1)。次に、監視スレッド24を生成する(S2)。そして、ポート22を作成する(S2)。続いて、サービススレッド群23を生成する(S4)。初期化スレッド26は、初期化処理が終了すると、自分自身を消滅させる(S5)。なお、本実施例では、初期化処理終了後、初期化スレッド26を消滅させるようにしているが、初期化スレッド26が、初期化処理終了後、監視スレッド24として動作するようにしてもよい。

【0040】図6は実施例におけるサービススレッド群

23の各サービススレッド23a~23cの処理手順を説明するフローチャートである。サービススレッド群23の各サービススレッド23a~23cは初期化スレッド26によって生成された後、クライアント21からのリクエスト28を処理する。

【0041】サービススレッド群23の各サービススレッド23a~23cはポート22を介してクライアント21からのリクエスト28を受信すると(S10)、リクエストされた処理のタイムアウト時間と自身のスレッドIDをタイムアウト表25に記入する(S11)。そして、リクエスト28の処理を開始する(S12)。該リクエスト28の処理が終了すると、クライアント21にその処理結果を返す(S13)。そして、タイムアウト表25から自身が記入したタイムアウト時間とスレッドIDを削除して(S14)、クライアントからの次のリクエスト28を待つ。

【0042】図7は実施例における監視スレッド24の処理手順を説明するフローチャートである。監視スレッド24は、初期化スレッド26によって生成された後、図7に示す手順でサービススレッド群23の動作を監視する。

【0043】監視スレッド24は、まず、マイクロカーネル27がシステムコールとして提供するタイマルーチンを用いて自身のタイムアウト時間を設定し、マイクロカーネル27に対してタイムアウト処理を依頼する(S20)。次に、タイムアウト表25からサービススレッド群23の各サービススレッド23a~23cが記入したタイムアウト時間を読み出す(S21)。該読み出したタイムアウト時間と現在の時刻を比較することによって、タイムアウトしたサービススレッドが存在するかを判断する(S22)。該判断によって、タイムアウトしたサービススレッドを検出した場合には(S22: YES)、該タイムアウトしたサービススレッドを削除する(S23)。そして、該削除されたサービススレッドが受信したリクエスト28の送信元クライアント21にエラーをポート22を介して通知する(S24)。監視スレッド24は、削除した数のサービススレッドを新規に生成して、サービススレッドを補充する(S25)。

【0044】該ステップS25の処理後、若しくは、上記ステップS22にてタイムアウトしたサービススレッドを検出なかった場合、監視スレッド24は一定時間スリープする(S26)。監視スレッド24のプライオリティは一般に高く設定されるため、スリープ状態にならないと、サービススレッド23a~23cが動作できなくなったり、監視処理によって必要以上に処理時間が消費されてしまう。また、このスリープ時間を調整することによって、監視スレッド24の定周期起動の周期、すなわち、サービススレッド23a~23cのタイムアウト監視処理の周期を変えられる。

【0045】上記スリープ時間が経過後、自身が設定したタイムアウト処理の解除をマイクロカーネル27へ依頼する(S27)。その後、監視スレッド24は再び、上記処理S20~S27を繰り返す。

【0046】図8は、本実施例のサーバ20の動作を、クライアント21とマイクロカーネル27の動作とあわせて説明する図である。まず、マイクロカーネル27によってサーバ20が起動されると(S100)、サーバ20は、初期化スレッド26を生成する。初期化スレッド26は、生成されると、タイムアウト表25を初期化する(S101)。次に、監視スレッド24の生成をマイクロカーネル27へ依頼する(S102)。該依頼を受けたマイクロカーネル27は、監視スレッド24を生成する(S103)。

【0047】続いて、初期化スレッド26は、リクエスト受付ポート22の作成をマイクロカーネル27へ依頼する(S104)。該依頼を受けたマイクロカーネル27は、リクエスト受付ポート22を作成する(S105)。そして、初期化スレッド26は、サービススレッド群23の生成をマイクロカーネル27へ依頼する(S106)。該依頼を受けたマイクロカーネル27は、サービススレッド23a~23cから成るサービススレッド群23を生成する(S107)。初期化処理が終了すると、初期化スレッド26は、自分自身、すなわち、初期化スレッド26の消滅をマイクロカーネル27へ依頼する(S108)。該依頼を受けたマイクロカーネル27は、初期化スレッド26を消滅させる(S109)。

【0048】クライアント21は、サーバ20に対してサービス処理を依頼する必要があると、該サービス処理を依頼するリクエストメッセージを、サーバ20のポート番号を指定して送信する(S110)。該メッセージを受信すると、マイクロカーネル27は、該メッセージをサーバ20のポート22が有する前記受信キュー30につなぐ(S111)。

【0049】サービススレッド群23の各サービススレッド23a~23cは、前記ステップS107にて生成されると、まず、サーバ20のポート22が有する前記受信キュー30からリクエストメッセージを読み出す(S112)。そして、自分のスレッドID及びサービス処理のタイムアウト時間をタイムアウト表25に記入する(S113)。タイムアウト表25への記入後、リクエストされたサービス処理を開始する(S114)。

【0050】該サービス処理が終了すると、サービススレッド23a~23cは、クライアント21へのリプライ29の送信をマイクロカーネル27へ依頼する。該依頼を受けたマイクロカーネル27は、リプライメッセージをクライアント21宛の前記送信キューにつなぐ処理を行う(S116)。クライアント21は、上記送信キューに格納されているリプライメッセージを受信する(S117)。

【0051】クライアント21へのリブライ処理が終了すると、サービススレッド23a~23cは、タイムアウト表25から自分が記入したエントリを削除する(S118)。そして、上記ステップS112~S118を繰り返す。

【0052】一方、監視スレッド24は、前記ステップS103にて生成されると、まず、自分自身のタイムアウト時間を設定し、タイムアウト処理をマイクロカーネル27へ依頼する(S119)。該依頼を受けたマイクロカーネル27は、監視スレッドに対してタイムアウト時間を設定する(S120)。そして、監視スレッド24は、タイムアウト表25を参照しタイムアウトしたサービススレッドがあるか否かを判別することによって、サービススレッド23a~23cの監視処理を行う(S121)。監視処理終了後、監視スレッド24は、所定時間スリープを行う(S122)。そして、前記ステップS119で依頼したタイムアウト処理の解除をマイクロカーネル27へ依頼する(S123)。該依頼を受けたマイクロカーネル27は、タイムアウト処理の解除処理を行う(S124)。

【0053】なお、上記実施例では、初期化スレッド26によって、予めサービススレッド23a~23cを生成しているが、リクエストが届いた段階で、メッセージハンドラー等によって必要なサービススレッドを動的に生成するようにしてもよい。

【0054】また、上記実施例においては、クライアントからの依頼をスレッドによって処理するようにしているが、スレッドの代わりにプロセスによって処理するようにすることも可能である。

【0055】また、サービススレッド23a~23cが、「タイムアウト時間」フィールド25bに所定の時間(タイムアウト時間)を設定し、監視スレッド24が、定期的に該時間から適当な時間を減算(デクリメント)し、結果が零より大きいかな否かによって、設定されたタイムアウト時間までに処理を終了していないサービススレッドを検出するようにしてもよい。

【0056】本発明は、上記実施例以外に、マルチプロセッサ環境での並列処理システムやネットワークで接続された複数の計算機から成る分散処理システム等に実装することも可能である。また、本発明は、マイクロカーネル方式のOSのみに限定されるものでもなく、それ以外のOSを用いたクライアント/サーバモデルのデータベースシステムなどの各種アプリケーションシステムなどにも適用可能なものである。

【0057】

10 【発明の効果】以上、詳細に説明したように、本発明によれば、複数のクライアントからのリクエストを並列に処理するために、サーバ内のサービススレッド数を増やしても、タイムアウト設定のためのシステムコールの実行回数は増加しないので、サーバのレスポンス性能の低下を防ぐことができる。

【図面の簡単な説明】

【図1】本発明の原理図(その1)である。

【図2】本発明の原理図(その2)である。

【図3】本発明の一実施例の概要を説明する図である。

20 【図4】タイムアウト表の一構成例を示す図である。

【図5】初期化スレッドの処理手順を説明するフローチャートである。

【図6】サービススレッドの処理手順を説明するフローチャートである。

【図7】監視スレッドの処理手順を説明するフローチャートである。

【図8】サーバ20の動作を、クライアント21とマイクロカーネル27の動作とあわせて説明する図である。

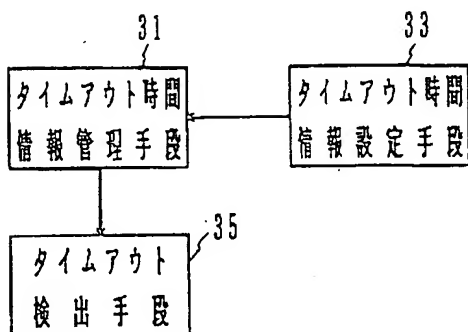
【図9】従来技術を説明する図である。

30 【符号の説明】

31	タイムアウト時間情報管理手段
33	タイムアウト時間情報設定手段
35	タイムアウト検出手段
100	サービス手段
101	タイムアウト時間管理手段
102	監視手段

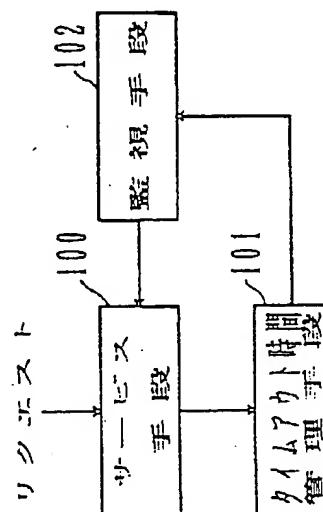
【图 1】

本発明の原理図（その１）



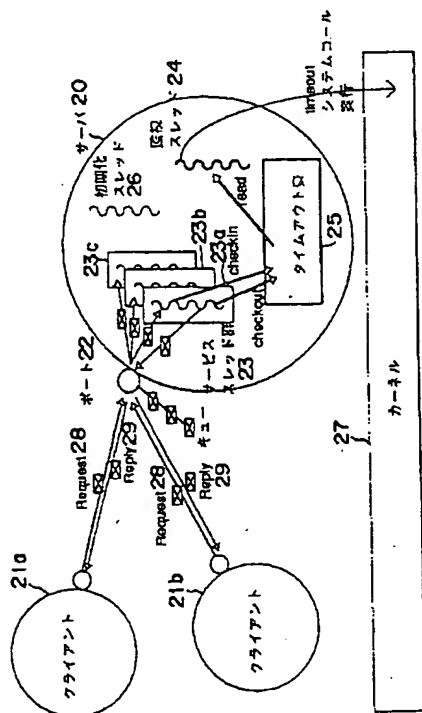
【图2】

本発明の原理図 (その2)



【図 3】

本発明の一実施例の概要を説明する図



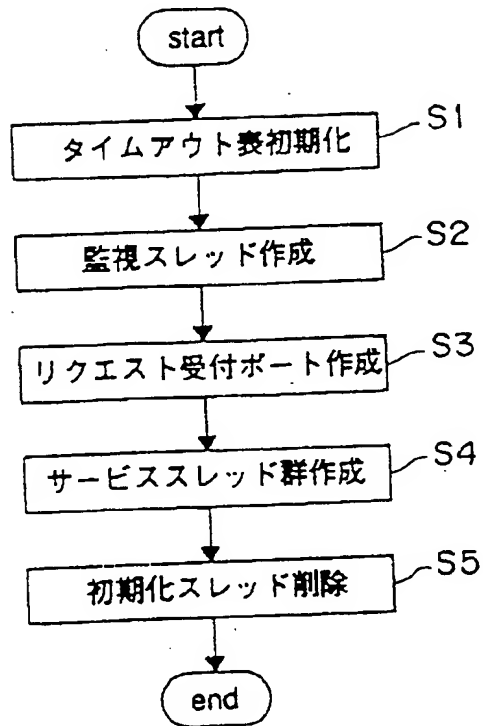
【図 4】

タイムアウト救の一構成例を示す図

[illegible]

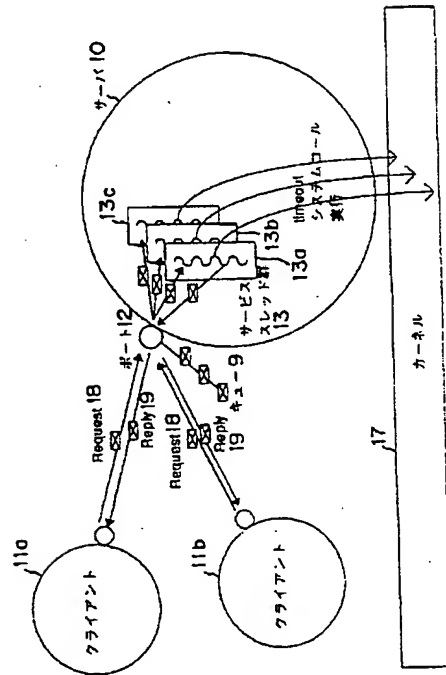
【図5】

初期化スレッドの処理手順を
説明するフローチャート



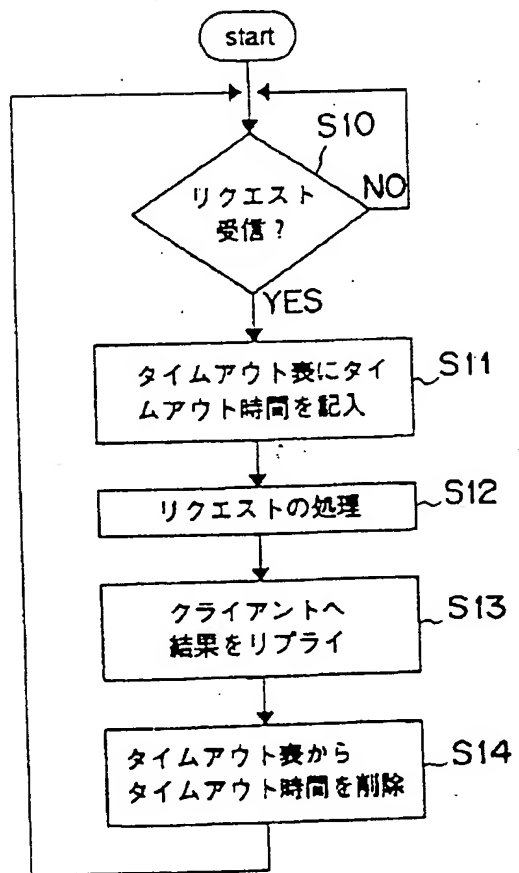
【図9】

従来の技術と説明する図



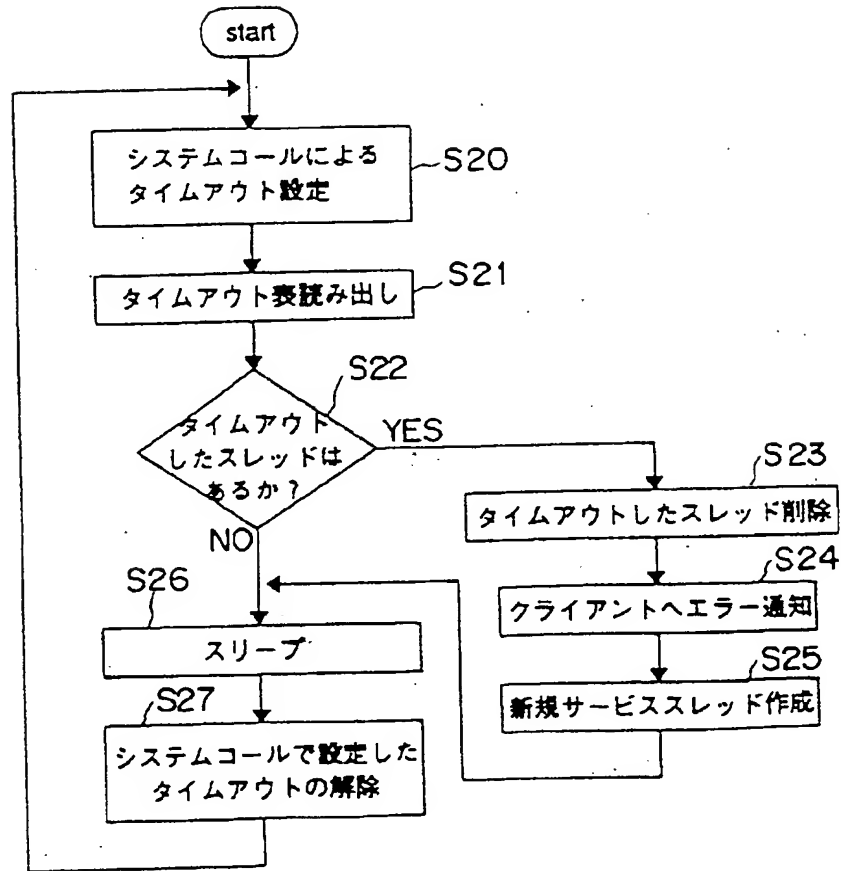
【図6】

サービススレッドの処理手順を
説明するフローチャート



【図7】

監視スレッドの処理手順を説明するフローチャート



【図8】

サーバ20の動作を、クライアント21とカーネル27の
動作とあわせて説明する図

